AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method for selectively monitoring load instructions to support transactional execution of a process, comprising:

starting a transactional execution of a block of instructions in a program, wherein starting the transactional execution involves executing an explicit ~~performing an~~ instruction implemented in hardware to start the transactional execution;

encountering a load instruction during the transactional execution, wherein changes made during the transactional execution are not committed to the architectural state of a processor until the transactional execution successfully completes;

determining whether the load instruction is a monitored load instruction or an unmonitored load instruction by analyzing the load instruction;

if the load instruction is a monitored load instruction,

performing a corresponding load operation, and

load-marking a cache line associated with the load instruction to facilitate subsequent detection of an interfering data access to the cache line from another process; and

if the load instruction is an unmonitored load instruction, performing the corresponding load operation without load-marking the cache line.

2. (Original) The method of claim 1, wherein prior to executing the program, the method further comprises generating the instructions for the program, wherein generating the instructions involves:

determining whether load operations that take place during transactional execution need to be monitored;

generating monitored load instructions for load operations that need to be monitored; and

generating unmonitored load instructions for load operations that do not need to be monitored.

3. (Original) The method of claim 2, wherein determining whether a load operation needs to be monitored can involve examining a data structure associated with the load operation to determine whether the data structure is a "protected" data structure for which loads need to be monitored, or an "unprotected" data structure for which loads do not need to be monitored.

4. (Original) The method of claim 2, wherein determining whether a load operation needs to be monitored can involve determining whether the load operation is directed to a heap, wherein loads from the heap need to be monitored and loads from outside the heap do not need to be monitored.

5. (Original) The method of claim 2, wherein determining whether a load operation needs to be monitored can involve allowing a programmer to determine if the load operation needs to be monitored.

6. (Currently amended) The method of claim 1, wherein determining whether the load instruction is a monitored load instruction involves examining an op code of the load instruction.

7. (Currently amended) The method of claim 1, wherein determining whether the load instruction is a monitored load instruction involves examining an address associated with the load instruction to determine whether the address falls within a range of addresses for which loads are monitored.

8. (Original) The method of claim 7, wherein examining the address involves comparing the address with one or more boundary registers.

9. (Original) The method of claim 7, wherein examining the address involves examining a Translation Lookaside Buffer (TLB) entry associated with the address.

10. (Original) The method of claim 1, wherein if an interfering data access from another process is encountered during transactional execution of the block of instructions, the method further comprises:

discarding changes made during the transactional execution; and

attempting to re-execute the block of instructions.

11. (Original) The method of claim 1, wherein if transactional execution of the block of instructions completes without encountering an interfering data access from another process, the method further comprises:

committing changes made during the transactional execution to the architectural state of the processor; and

resuming normal non-transactional execution of the program past the block of instructions.

12. (Original) The method of claim 1, wherein an interfering data access can include:

a store by another process to a cache line that has been load-marked by the process; and

a load or a store by another process to a cache line that has been store-marked by the process.

13. (Original) The method of claim 1, wherein the cache line is load-marked in level 1 (L1) cache.

14. (Currently amended) An apparatus that selectively monitors load instructions to support transactional execution of a process, comprising:

a start transactional execution mechanism configured to start a transactional execution of a block of instructions in a program, wherein starting the transactional execution involves executing an explicit ~~performing an~~ instruction implemented in hardware to start the transactional execution;

an execution mechanism within a processor;

wherein the execution mechanism is configured to support the transactional execution, and wherein changes made during the transactional execution are not committed to the architectural state of a processor until the transactional execution successfully completes;

wherein upon encountering a load instruction during transactional execution, the execution mechanism is configured to,

determine whether the load instruction is a monitored load instruction or an unmonitored load instruction by analyzing the load instruction,

if the load instruction is a monitored load instruction, to perform a corresponding load operation, and to load-mark a cache line associated with the load instruction to facilitate subsequent

detection of an interfering data access to the cache line from another process; and

if the load instruction is an unmonitored load instruction, to perform the corresponding load operation without load-marking the cache line.

15. (Original) The apparatus of claim 14, further comprising an instruction generation mechanism configured to:

determine whether load operations that take place during transactional execution need to be monitored;

generate monitored load instructions for load operations that need to be monitored; and to

generate unmonitored load instructions for load operations that do not need to be monitored.

16. (Original) The apparatus of claim 15, wherein the instruction generation mechanism is configured to determine whether a load operation needs to be monitored by examining a data structure associated with the load operation to determine whether the data structure is a "protected" data structure for which loads need to be monitored, or an "unprotected" data structure for which loads do not need to be monitored.

17. (Original) The apparatus of claim 15, wherein the instruction generation mechanism is configured to determine whether a load operation needs to be monitored by determining whether the load operation is directed to a heap, wherein loads from the heap need to be monitored and loads from outside the heap do not need to be monitored.

18. (Original) The apparatus of claim 15, wherein the instruction generation mechanism is configured to determine whether a load operation needs to be monitored by allowing a programmer to determine if the load operation needs to be monitored.

19. (Original) The apparatus of claim 14, wherein the execution mechanism is configured to determine whether the load instruction is a monitored load instruction by examining an op code of the load instruction.

20. (Original) The apparatus of claim 14, wherein the execution mechanism is configured to determine whether the load instruction is a monitored load instruction by examining an address associated with the load instruction to determine whether the address falls within a range of addresses for which loads are monitored.

21. (Original) The apparatus of claim 20, wherein the execution mechanism is configured to examine the address by comparing the address with one or more boundary registers.

22. (Original) The apparatus of claim 20, wherein the execution mechanism is configured to examine the address by examining a Translation Lookaside Buffer (TLB) entry associated with the address.

23. (Original) The apparatus of claim 14, wherein if an interfering data access from another process is encountered during transactional execution of the block of instructions, the execution mechanism is configured to:

discard changes made during the transactional execution; and to

attempt to re-execute the block of instructions.

24. (Original) The apparatus of claim 14, wherein if transactional execution of the block of instructions completes without encountering an interfering data access from another process, the execution mechanism is configured to:

commit changes made during the transactional execution to the architectural state of the processor; and to

resume normal non-transactional execution of the program past the block of instructions.

25. (Original) The apparatus of claim 14, wherein an interfering data access can include:

a store by another process to a cache line that has been load-marked by the process; and

a load or a store by another process to a cache line that has been store-marked by the process.

26. (Original) The apparatus of claim 14, wherein the cache line is load-marked in level 1 (L1) cache.

27. (Currently amended) An computer system that selectively monitors load instructions to support transactional execution of a process, comprising:

a processor;

a memory;

a start transactional execution mechanism within the processor configured to start a transactional execution of a block of instructions in a program, wherein starting the transactional execution involves executing an explicit ~~performing an~~ instruction implemented in hardware to start the transactional execution;

an execution mechanism within the processor;

wherein the execution mechanism is configured to support the transactional execution, and wherein changes made during the transactional execution are not committed to the architectural state of a processor until the transactional execution successfully completes;

wherein upon encountering a load instruction during transactional execution, the execution mechanism is configured to,

determine whether the load instruction is a monitored load instruction or an unmonitored load instruction by analyzing the load instruction,

if the load instruction is a monitored load instruction, to perform a corresponding load operation, and to load-mark a cache line associated with the load instruction to facilitate subsequent detection of an interfering data access to the cache line from another process; and

if the load instruction is an unmonitored load instruction, to perform the corresponding load operation without load-marking the cache line.